FR•
Free Radical Labs

Observing the Temperature of the TMP36 Temperature Sensor

By now you've learned how to upload code, wire components onto a breadboard and control an LED with an analog input; you've come a long way! This tutorial will have you observe, on your computer screen, the temperature in the room using the TMP36 temperature sensor. By the end, you'll know how to read a datasheet and observe variables directly using the Serial class of functions.

Part 1. Gathering Materials

To complete this tutorial, you'll need a few things:

- A computer with the Arduino IDE installed and setup for the Sparkfun Redboard
- 1 x TMP36 temperature sensor
- 3 x 6" Jumper Wire
- 1 x breadboard
- 1 x Sparkfun Redboard
- 1 x USB (B) Cable

Part 2. Wiring the Components

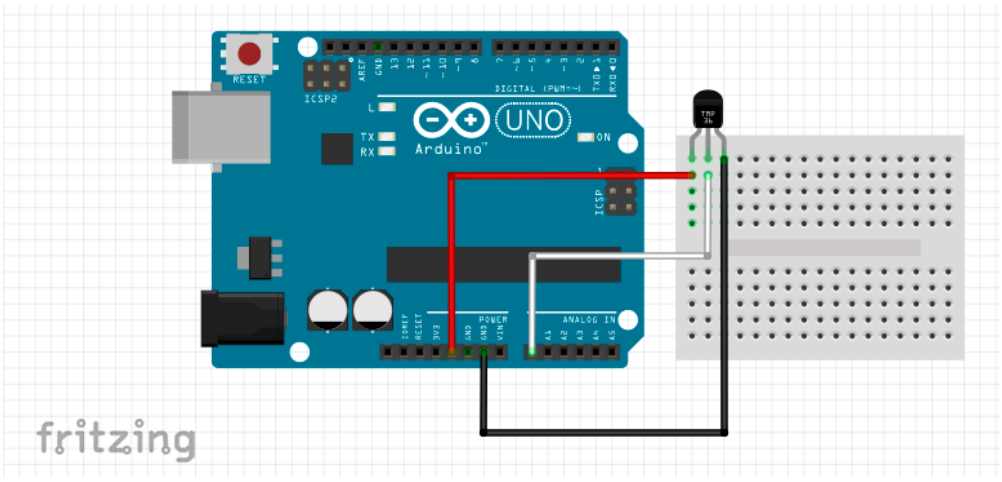

Figure 1. Wiring Diagram for the TMP36 temperature sensor

After acquiring the parts, wire the sensor to how it's shown in figure 1. While wiring, make sure that the flat side of the sensor is facing you. The leftmost pin is connected to 5V and the rightmost pin is connected to GND. The middle pin contains the data for the temperature and will connect to the A0 analog input. Learning to wire the sensor is as easy as reading the

datasheet (http://cdn.sparkfun.com/datasheets/Sensors/Temp/TMP35_36_37.pdf). Figure 4 in that datasheet shows how this sensor is to be wired.

Part 3. Reading a Datasheet

Unfortunately when you get a new sensor in the mail, like the TMP36, it doesn't come with an instruction manual. The first thing to do is to lookup the datasheet online. This will show how to wire it, what voltages are appropriate for the device, the functional diagram for how it works, how to use it in a program, and many other things. Thankfully this is a relatively simple sensor to interface with and doesn't require hours of reading like the MLX90620 temperature sensor. For this application, we are mostly concerned with Figure 6 in the datasheet, shown in this tutorial in Figure 2.

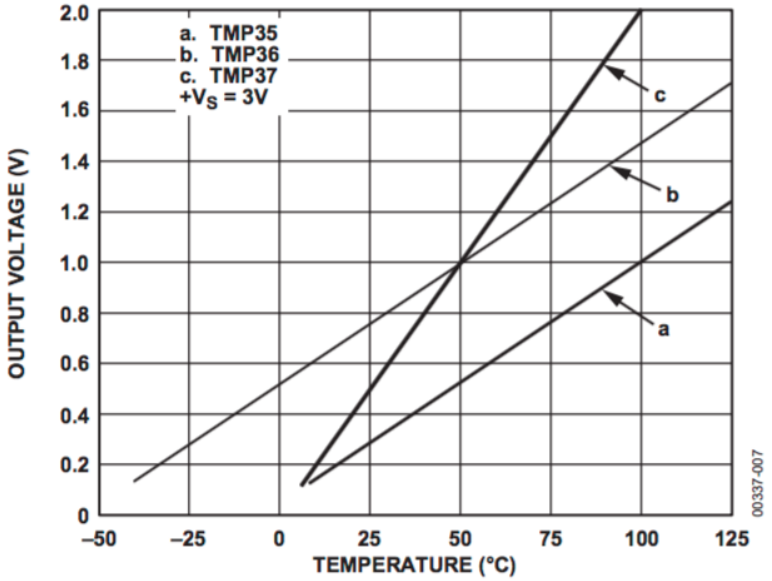

Figure 6. Output Voltage vs. Temperature

Figure 2. Output voltage versus temperature (°C) for the TMP family of temperature sensors

We'll be using the TMP35 sensor, Figure 2 shows that, for the line labeled (b), at 0°C we should expect the output voltage to be 0.5V and at 50°C we expect the output to be 1V. This shows that for each degree Celsius increase, we should expect an increase of 10 mV on the output. Because this is a linear relationship with a slope of 10mV / °C, the equation for the line is:

$$\text{Temperature (°C)} = (\text{Output Voltage} - 0.5)*100$$

To convert from Celsius to Fahrenheit, we will use the following relationship:

$$°F = (1.8 * °C) + 32$$

Now we're ready to program the TMP36 temperature sensor.

Part 4. Programming

```
Void setup(){
/* This initiates the connection for the
  * arduino to communicate back data to
  * the computer at a bitrate equal to the
  * number inside the parenthesis. The Arduino
  * IDE and most other programmers use 9600 bits/sec
  * as a default.
  */
 Serial.begin(9600);

}

void loop() {

  /* First, we'll gather the data that the
   * temperature sensor is providing as
   * an output
   */

  int rawValue = analogRead(A0);

  /* The datasheet correlates voltage to temperature.
   *  We'll first need to convert the analogRead into
   *  an actual voltage. Knowing that 0V corresponds to
   *  a reading of 0 and a reading of 5V corresponds to
   *  a reading of 1023 means that they're related by:
   *
   *  Voltage = (5/1023)*analogRead
   *
   *  Another thing to note here is the difference between
   *  different types of number values.
   *
   *  int type: if a variable is declared as an int, it can
   *  only be a whole number between -32,678 and +32,677.
   *  The use of this variable uses 2 bytes of information
   *
   *  float type: if a variable is declared as a float, it
   *  can be between -3.402824E+38 and +3.402824E+38 with 6
   *  digits of decimal space available. For numbers that may
   *  contain decimals, like in this tutorial, we will want
   *  to use a float. The use of this variable uses 4 bytes
   *  of information
   *
   *  One last point to note is how division works in Arduino.
   *  Theres whole number division, and decimal precision division.
   *
   *  For example, 5/1023 will result in zero because it rounds down
   *  to the nearest whole number. While 5.0/1023.0 will result
```

```
 *  in a more exact value of 0.004888
 *
 */

 float voltageReading = (5.0/1023.0) * rawValue;

//Using the conversion of:
//Temperature = (Voltage - 0.5)*100

 float tempCelsius = (voltageReading - 0.5)*100.0;

 //Using the conversion of:
 //Fahrenheit = 1.8 * Celsius + 32

 float tempFahrenheit = (tempCelsius * 1.8) + 32;

 /* Now we'll want to print out the values of these variables
  *  and see how the sensor responds when you breathe on it!
  *
  *  To print a line of text, we'll use Serial.print() and
  *  Serial.println() functions and put
  *  what we want inside the parenthesis, with quotes surrounding
  *  the text. Or no quotes in the case of just printing a variable
  *
  *  If you'd like to print multiple things on the same line,
  *  Serial.print() allows you to do that.
  *
  *  For the following code:
  *
  *  int test = 5;
  *  Serial.println("Hello World");
  *  Serial.println(test);
  *  Serial.print("Hello World");
  *  Serial.print(test);
  *
  *  Will print the following inside the Serial Monitor:
  *
  *  Hello World
  *  5
  *  Hello World5
  *
  */

  Serial.print("The Voltage on the Input is: ");
  Serial.print(voltageReading);
  Serial.print(" V");
  Serial.println();

  Serial.print("The Temperature is: ");
  Serial.print(tempCelsius);
  Serial.print(" degree(s) Celsius");
```

```
  Serial.println();

  Serial.print("Or: ");
  Serial.print(tempFahrenheit);
  Serial.print(" degree(s) Fahrenheit");
  Serial.println();
  Serial.println();

  /* Using the delay() command, we can control
   *  how quickly the Arduino reads the temperature
   *  in the room. The number inside the parenthesis
   *  is the wait time, in milliseconds
   *
   */
  delay(500);

}
```

Part 5. Uploading the program and testing

Now, upload the code and click on the magnifier glass in the top right window. A new window should appear and start printing lines of text. The output should look like something in Figure 3, and print a new section every half second.

```
The Voltage on the Input is: 0.65 V
The Temperature is: 15.00 degree(s) Celsius
Or: 59.01 degree(s) Fahrenheit
```

Figure 3. Sample Output

Let the program run for a few seconds and you should start to see that the output is the same every time it prints. Now breathe onto it! The warm air from your breath should cause the temperature sensor to start increasing in temperature, and if left alone for some time, will return to the value it was at before the breath.

That concludes this tutorial, introducing you to how to interface with a sensor, read a datasheet, and view its output onto a computer monitor. The next one will show you how to create a night light with the light sensor and an LED

Happy Tinkering!